IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Trusted Operating System Approach for Security of an OS with JAVA

Gaurav Indoria[*1], **Kamlesh Lakhwani**[2]

[*1,2]Suresh Gyan Vihar University, Jaipur, Rajasthan, India

gauravindoria@gmail.com

Abstract

In a little more than a decade since the inception of the Java language, the Java ecosystem has evolved into one that businesses around the world look to for secure, robust, and manageable web-based applications that serve the enterprise. Java was created to be exploited in highly distributed environments. The portability of the language meant that security had to be incorporated in the very first steps of Java application development. This way, application users could be assured that the code they obtained was secure and had not been tampered with during transmission. This paper explains how to achieve that security. This paper explores the integrated Java platform security that results from the synergy of these two world-class environments. Java security as implemented by the JVM operates above the operating system security services, meaning that Java Security is actually implemented in the JVM run-time environment. The privileges to be given at the host operating system level to the Java applications execution streams are well established and fully controlled. The paper describes the Java security model as implemented in the z/OS JVM™. In this paper we are also discussing about various security levels of an OS. This paper will give the brief description of RACF & its groups with respect to z/OS. In this paper we are trying to integrate the JAVA security with the security of an OS. This paper will explain about the OS Security Environment, its terminology, RACF, JNI & Security, & Trusted Operating System.

General Terms: Security, Trusted OS, JAVA, JNI, Levels.

Keywords: Java, Java Security, JVM, z/OS, Security Levels, Trusted OS

Introduction

Operating system: Collection of programs that allows user to operate computer hardware. An OS generally has three layers: Inner-layer, Middle-layer, Outer-layer. A small system might have three to five users, and a large system might have several thousand users. Some installations have all their workstations in a single, relatively secure, area. Others have widely distributed users, including users who connect by dialing in and indirect users connected through personal computers or system networks. You need to understand the features and options available so that you can adapt them to your own security requirements. Flaws in the operating systems of computers are discovered almost daily. The majority of viruses take advantage of these flaws to infect your computer. Once a virus enters your system, it can potentially cause devastating damage. To avoid contracting a virus, we should take the Software Updates, Firewall, Account Management, Antivirus Software. System security is often associated with external threats, such as hackers or business rivals. However, protection against system accidents by authorized system users is often the greatest benefit of a well-designed security system. In a system without good security features, pressing the wrong key might result in deleting important information. System security can prevent this type of accident. The best security system functions cannot produce good results without good planning. Security that is set up in small pieces, without planning, can be confusing. It is difficult to maintain and to audit. Planning does not imply designing the security for every file, program, and device in advance. It does imply establishing an overall approach to security on the system and communicating that approach to application designers, programmers, and system users.

System security has three important objectives:

Confidentiality:

- Protecting against disclosing information to unauthorized people.

- Restricting access to confidential information
- Protecting against curious system users and outsiders

Integrity:
- Protecting against unauthorized changes to data
- Restricting manipulation of data to authorized programs
- Providing assurance that data is trustworthy

Availability:
- Preventing accidental changes or destruction of data
- Protecting against attempts by outsiders to abuse or destroy system resources

The Operating System security Environment

Main component of operating system security environment are as follows:

Services:- Used to gain access to the OS and its features. It Includes: user authentication, remote access administration tasks, password policies.

Files:- Common threats: File permission, File sharing. Files must be protected from unauthorized reading and writing actions. Data resides in files; protecting files protects data, Read, write, and execute privileges.

FTP (File Transfer Protocol):- Internet service for transferring files from one computer to another, Transmits usernames and passwords in plaintext, Root account cannot be used with FTP, Anonymous FTP: ability to log on to the FTP server without being authenticated.

Sharing of Files:- Naturally leads to security risks and threats, Peer-to-peer programs: allow users to share files over the Internet, Reasons for blocking file sharing:
(i) Malicious code (ii) Adware and spyware
(iii) Privacy and confidentiality
(iv) Porno-graphy (v) Copyright issues

Memory:- Hardware memory available on the system, that Can be corrupted by badly written software so there are two options: either we should stop using the program or apply a patch (service pack) to fix it. It can harm data integrity and can potentially exploit data for illegal use.

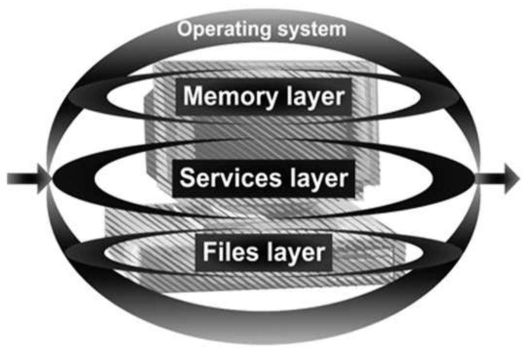

Figure 1: Operating System Security Environment

Basic Terminology

This topic provides users with basic security terminology.

Object: An object is a named space on the system that you or an application can manipulate. Everything on the system that you or an application can work with is considered an object. Objects provide a common interface for working with system components. The most common examples of objects are files and programs. Other types of objects include commands, queues, libraries, and folders. Objects on the system are identified by object name, object type, and the library in which the object resides. You can secure each object on the system.

Library: A library is a special type of object that is used to group other objects. Many objects on the system reside in a library. Libraries are essentially containers, or organizational structures for other objects, and you can use them to reference other objects on your system. Libraries might contain many objects, and might be associated with a specific user profile or application. QSYS, which contains all other libraries on the system, is the only library that can contain other libraries. Objects in a library are handled like objects in a subdirectory. A library cannot live inside a directory.

Directory: A directory is a special object that provides another way to group objects on the system. Objects can reside in a directory and a directory can reside in another directory, forming a hierarchical structure. Each file system is a major *subtree* in the integrated file system directory structure. Directories are different from libraries in that the address of each library maps to the QSYS library while directories are not addressable. Names of libraries are restricted to 10 characters while directories can have longer names which might be case sensitive. Directories can have multiple names because the path to the directory is what is named and not the directory itself. You can use different commands and authority requirements when working with directories and libraries.

User profile: Every system user must have a user identity before they can sign on to and use a system. This user identity is a special object called a user profile, which only an administrator with appropriate system authority can create for a user.

Special authority: Special authority determines whether the user is allowed to perform system functions, such as creating user profiles or changing the jobs of other users.

Physical security: Physical security includes protecting the system unit, system devices, and backup media from accidental or deliberate damage. Most measures you take to ensure the physical security of your system are external to the system. Certain system models are equipped with a keylock that prevents unauthorized functions at the system unit.

Application security: Application security deals with the applications you store on your system and how you will protect those applications while simultaneously allowing users access to them.

Resource security: Resource security on the system allows you to define who can use objects and how objects can be used. The ability to access an object is called *authority*. When you set up object authority, you need to be careful to give your users enough authority to do their work without giving them the ability to browse and change the system. Object authority gives permissions to the user for a specific object and can specify what the user is allowed to do with the object. An object resource can be limited through specific, detailed user authorities such as adding records or changing records. System resources can be used to give the user access to specific system-defined subsets of authorities: *ALL, *CHANGE, *USE, and *EXCLUDE. System values and user profiles control who has access to your system and prevent unauthorized users from signing on. Resource security controls the actions that authorized system users can perform, and the objects that they can access after they have signed on successfully. Resource security supports the main goals of security on your system to protect:

- Confidentiality of information
- Accuracy of information to prevent unauthorized changes
- Availability of information to prevent accidental or deliberate damage

Security policy: A security policy allows you to manage security on an i5/OS® system. Use the eServer™ Security Planner to help you plan for and carry out a basic security policy for your servers.

Security Levels

The Operating System platform offers five levels of security:-

Level 10: Password security
At security level 10, you have no security protection. Therefore, security level 10 is not recommended.

Level 20: Password security
At this security level, users who need to access the system must have a password and user ID that the system recognizes. The system administrator creates both the user ID and initial password for users. This level of security allows users total authority to do anything they want on the system, which means that all users can access all data, files, objects, and so on, on your system because all users have *ALLJOB special authority.

Level 30: Password and resource security
Level 30 provides more security functions in addition to what is provided at level 20. Users must have specific authority to use resources on the system. Users do not have automatic access to everything on the system and the system administrator must define a valid user ID and password for them. User access is limited by the security policies of the business.

Level 40: Integrity protection
At this security level, resource security and integrity protection are enforced, and the system itself is protected against users. Integrity protection functions, such as the validation of parameters for interfaces to the operating system, help protect your system and the objects on it from tampering by experienced system users. For example, user-written programs cannot directly access the internal control blocks through pointer manipulation. Level 40 is the default security level for every new installation and is the recommended security level for most installations.

Level 50: Advanced integrity protection
At this security level, advanced integrity protection is added to the resource security and level 40 integrity protection enforcement. Advanced integrity

protection includes further restrictions, such as the restriction of message-handling between system state programs and user state programs. Not only is the system protected against user-written programs, but it ensures that users only have access to data on the system, rather than information about the system itself. This offers greater security against anyone attempting to learn about your system. Level 50 is the recommended level of security for most businesses, because it offers the highest level of security currently possible. Also, level 50 is the required level for C2, FIPS-140, and Common Criteria certifications.

Java And Security

The Java platform offers many different APIs to interact with various system resources. However, there are still situations in which services of the hosting platform cannot be invoked using the provided Java API. In other words, this is a case where a given application cannot be written entirely in Java. For such applications, the Java Native Interface (JNI™) framework might provide a solution. The JNI can be used in two ways:

- It can be used to implement native methods in a Java application to invoke non-Java functions.
- It can also be used to modify a native application so that it can be called by Java applications.

JNI was originally designed to be used in combination with C/C++. However Enterprise COBOL for z/OS V3R4 proposes now inter-language interoperability using JNI. In most practical cases today, JNI is still used in combination with C/C++. In the z/OS case, C/C++ allows z/OS users to access a large number of z/OS UNIX Assembler Callable Services. In turn, this enables these users to reach resources that may not be accessible directly through Java code. This is a situation where users could decide to use a native language to achieve their goals.

Figure 2 illustrates an example of a Java program that uses JNI and C to get to a z/OS resource. This infrastructure comprises essentially three functions:

1. A Java program that is the driver which writes to the output device. The source code should contain the interaction between Java and the native method invocation. And preferably, it should also contain logic to deal with error conditions.
2. A Java class with the native method definition.
3. A DLL module, written in C, that contains the entry points for the basic functions. Those functions are then mapped by the z/OS C/C++ Run Time Library (RTL) to Assembler Callable Services (BPX....), which in turn are executed by the z/OS UNIX Kernel. These callable services allow z/OS UNIX programs to access system resources.

Therefore, the JNI is basically located between the Java Virtual Machine (JVM) and the operating system (z/OS). However, instead of going directly from the JVM to the operating system (as the dotted line in Figure 1-1 indicates), a native C/C++ library is used to invoke access to the required resources.
Using JNI offers advantages over using a 100% pure Java program. For example, it allows users to consider the reuse of code from another language that is already available. In addition, the native code generally executes faster than Java code, so some performance gains could also be expected from the use of JNI.

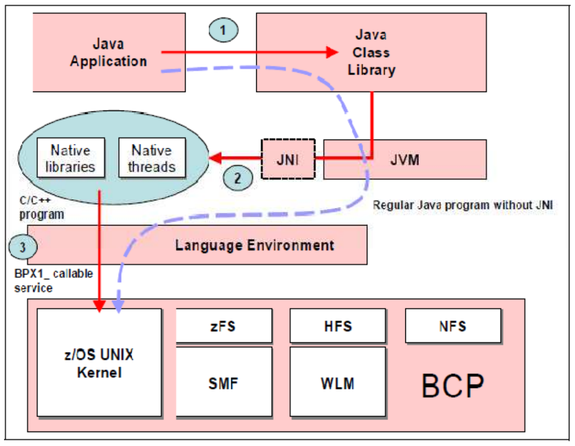

Figure 2: Example of using native methods for file I/O

However, there are also disadvantages to using JNI that users need to consider:

- Using JNI is not recommended for a J2EE™ environment such as Web-Sphere Application Server, because any operational anomalies in the calling program adversely affect the operations of the J2EE server.
- The zAAP (z Application Assist Processor) engine can exclusively be used to run Java code. Therefore, to realize substantial benefit from the zAAP, it is recommended that the use of native code be somehow limited.

Resource Access Control Facility (RACF)

The Multiple Virtual Storage (MVS) operating system was designed in the 1970s to run on the System/370™ (S/370™), for the secure execution of multiple applications on one system by multiple users. RACF was introduced in 1976 to work on top of the robust hardware and operating system layers to provide applications with a common point of user authentication and access control services. At that point it was termed an “External Security Manager”, because security management was becoming “external” to the applications.

Since then, RACF has grown and been adapted to meet the specific needs and challenges of new security technologies supported by the OS/390 and z/OS operating systems. As of z/OS V1R10, RACF provides support for the following security services:

_User identification and authentication for z/OS users via password, Pass-Ticket or password-phrase
_User identity mapping for z/OS users authenticated via X.509 V3 digital certificate or Kerberos ticket
_Resource access control for z/OS applications and components on the basis of the user’s identity or “groups of users” membership
_ X.509 digital certificate management
_ Kerberos user registry
_Remote security services via the z/OS LDAP server
_Support of the J2EE role security model
_Auditing of all security events detected by RACF or reported to RACF.
The title (Helvetica 18-point bold), authors' names (Helvetica 12-point) and affiliations (Helvetica 10-point) run across the full width of the page – one column wide. We also recommend e-mail address (Helvetica 12-point). See the top of this page for three addresses. If only one address is needed, center all address text. For two addresses, use two centered tabs, and so on. For three authors, you may have to improvise.

RACF Group

The RACF Security Administrators Guide describes five different "types" of groups and makes recommendations for their use. The types include:

- Administrative groups
- Holding groups
- Data-control groups
- Functional groups
- User groups

We tend to think of the first three types of RACF groups in generic terms as "structural" and the remaining as "access granting." We’ll explore each group in more detail.

Administrative groups:- These are any RACF groups created purely as an administrative convenience. Often, they're best used to describe functional areas of the business served by the z/OS system. Administrative groups usually have sub-groups, which they own, and aren't usually granted direct access to data or functions. User IDs are connected to the group with group-level privileges to administer the resources within the group-scope.

Holding groups:- Holding groups can serve many purposes and are often used as temporary storage for user IDs during some process - for example, moving a user ID to a new part of the RACF tree. Sometimes, they're useful as temporary owners of resources apart from user IDs. Holding groups aren't typically granted access to resources and don't usually have any sub-groups, being at the bottom of their branch of the group tree.

Data-control groups:- These groups are used as the High Level Qualifier (HLQ) for z/OS datasets. It's a requirement that all Dataset HLQs exist as either a user ID or a group in RACF before the dataset profiles can be created for that HLQ. Data-holding groups can be used to confer administrative rights over the dataset profiles they prefix. This is most often done by making the HLQ group the owner of the RACF dataset profiles beginning with that HLQ. Depending on the delegation strategy in use, user ID may or may not be connected to these groups. More on this topic will appear in a later article.

Functional groups:- These are the groups that actually grant access to data and resources via their presence in profile-access lists. As an administrative convenience, functional groups prevent the need to maintain profile-access lists containing extensive lists of user ID that require access to the resources. Instead, the functional group is added to the profile-access list, and the administrative activity now consists of maintaining the list of user ID in each functional group, rather than maintaining the profile-access list.

User groups:- Subtly distinct from functional groups, user groups can be used to collectively manage the access of diverse groups of user ID. A good example of this being the requirement for many users to access a wide variety of resources involved in the process of logging onto an application such as TSO or CICS*.
We'll re-examine the use of all these types of groups later; for now we need to know a little more about the relationship between user IDs and groups.

User IDs & GROUPS

User IDs must, by definition, be a member of at least one group, known as their "default group." There are no special implications in a group being a user ID's default group. Several applications - including CICS and TSO - provide the capability for a user ID to "switch" to another group to use as their default during the logon process. Of course, the user ID must already be a member of the group intended to be set as their default group for their next sign-on.

A little history may help explain the reason for this behavior. Changing your default group is rarely done on a modern RACF system. In early versions of RACF, however, users received access via their default group only, necessitating the need to switch their default group to access different sets of resources. Typically, a user ID was connected to a few groups, each representing some job function, and the user would choose the appropriate group at logon time depending on their intended work and the resource-access requirements for their next sign-on.

This changed with the introduction many years ago of the RACF "list of groups checking" feature that most modern installations employ by default. With list of groups active, a user receives access to any resource to which any of their groups have permission.

Another characteristic of user IDs is that they must - like all RACF profiles - be owned by something. A user ID may be owned by either another user ID or a group. In practice, it's strongly recommended to have all user ID owned by groups. The reason for this recommendation is the previously mentioned administrative rights that are conferred over owned resources. If a user ID owns another user ID, they may issue certain RACF commands to manipulate the owned user ID. Usually, this is not desirable.

So we see there is a necessity in RACF for a user ID to be associated with at least one group - the default group, and possibly one more - the owning group. Much philosophical debate exists among RACF specialists about the best use of these two unavoidable groups. The two most common configurations are:

- Default group and owning group are the same group
- Default group is a functional or user group; owning group is an administrative or holding group

User IDs can receive differing levels of privilege and administrative rights by setting certain options when they are connected to a RACF group. These options are referred to as group-connection attributes and group authorities.

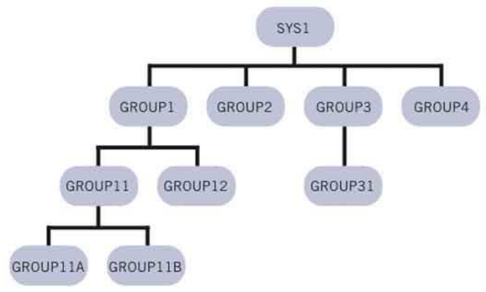

Figure 3: User IDs & Groups

Group Privileges Attributes & Authorities

As mentioned previously, RACF administrative rights can be conferred over owned resources. This is one reason to maintain the integrity of the ownership chain in the group tree. I'll briefly discuss some of these privileges, first covering the correct terminology to use.

When a user ID is connected to a RACF group, two sets of optional privileges may also be associated with the connection. The group connection attributes and the group authority. Listed from the highest privilege down, possible group connection attributes are group-SPECIAL, group-OPERATIONS and group-AUDITOR. Group authorities, in increasing level of privilege, are USE, CREATE, CONNECT and JOIN. To fully cover the possible interactions between these group privileges and other user ID-related settings is beyond the scope of this article. I'll instead list some of the most common configurations in practical use.

Provided that the group ownership chain is intact, group-SPECIAL is the most commonly employed method of delegating access to additional RACF administrative staff. Group- SPECIAL can be used with great effect to provide specific administrators the ability to manage RACF groups, resources and user IDs within a limited scope (i.e., branch) of the group tree. Figure 2, shows an example of the use of group attributes and is described below.

Staff involved in user administration would be connected to the group USERS with the group-SPECIAL attribute. They would be connected to group SALES or HR instead to restrict their scope of ownership.

Staff involved with system maintenance might be granted group-AUDITOR over the SYSTEM group or sub-groups, and staff administering access to applications would have the group-SPECIAL attribute over the APPS group or sub-groups. Group connection authority of SPECIAL would allow the user administrators access to manage user IDs, and authority of AUDITOR would allow the systems staff to list RACF information for the profiles under their scope (i.e., the SYSTEM group). Similarly, application administrators may also be granted authorities over one or many applications systems-related profiles owned by the group tree under their scope.

Group-SPECIAL confers more authority than is needed to allow simple connecting of user IDs to groups, which is where the Group authorities become useful.

Group authority USE is the default and allows access to resources via the group. Group authority CREATE allows the creation of new dataset profiles with that group as high-level qualifier - this is rarely used in practice. Authorities CONNECT and JOIN allow the connection and removal of user IDs from groups. JOIN is used in combination with class authority to create new user IDs.

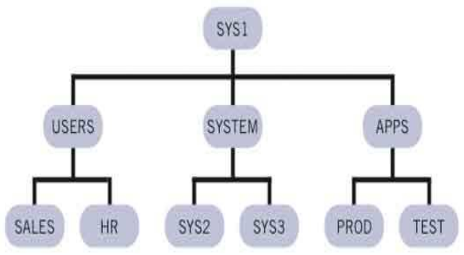

Figure 4: User IDs & Groups

Approch To OS security solution Trusted OS

Trusted operating systems (OS) provide the basic security mechanisms and services that allow a computer system to protect, distinguish, and separate classified data. Trusted operating systems have been developed since the early 1980s and began to receive National Security Agency (NSA) evaluation in 1984. Trusted OS may lower the security risk and the threat to the security holes of implementing a system that processes classified data. Trusted OS can implement some security policies and accountability mechanisms in an OS package via integrated or micro kernel type. A security policy is the rules and practices that determine how sensitive information is managed, protected, and distributed. Accountability mechanisms are the means of identifying and tracing who has had access to what data on the system so they can be held accountable for their actions.

In these days, the trusted OS is not used widely as a commercial purpose. But the researches about trusted OS are proceeding over the world, and new product type using the loadable security kernel module (LSKM) or dynamic link library (DLL) is being developed and some of such products are introduced. Most important concept in this paper is Security Level proposed by Dr. Tai-hoon Kim, because this concept should be considered in every operational environment.

Trusted OS may be used to implement mandatory access control (MAC) via multi-level security (MLS) systems and to build security countermeasures that allow systems of different security levels to be connected to exchange mutual data. Using of a trusted OS may be the way that a system can be connected to other high security systems. Department of Defense (DoD) security regulations define what evaluation criteria must be satisfied for a multi-level system based on the lowest and highest classification of the data in a system and the clearance level of the users of the system. Using an NCSC-evaluated system reduces accreditation cost and risk. The security officer identified as the Designated Approving Authority (DAA) for secure computer systems has the responsibility and authority to review and approve the systems to process classified information. The DAA will require analysis and tests of the system to assure that it will operate securely. The DAA can accept the NCSC evaluation of a system rather than generating the data. For a B3 or A1 system, which can represent a saving of 1 to 2 years in schedule and the operating system, will provide a proven set of functions.This technology has been implemented by several vendors for commercial-off-the-shelf (COTS) use in secure systems. As of September 1996, the NCSC Evaluated Product List indicated that fourteen OS have been evaluated as level C2, B1, B2, and B3 systems in the last three years [3]. The number of OS evaluated by class (excluding evaluations of updated versions of OS) is included in the table 1 [4]. Using of one of the approved trusted OS can result in substantial cost and schedule reductions for a system development effort and provide assurance that the system can be operated securely.

The heavy access control and accounting associated with high security systems can affect system performance; as such, higher performance processors, I/O, and interfaces may be required. Trusted OS have unique interfaces and operating controls that require special security knowledge to use and operate. Frequently COTS products that operate satisfactorily with a standard operating system must be replaced or augmented to operate with a trusted operating.

Table 1: NCSC Evaluation Criteria Classes.

Class	Title
A1	Verified Design
B3	Security Domains
B2	Structured Protection
B1	Labeled Security Protection
C2	Controlled Access Protection
C1	Discretionary

Conclusion And Future Work

Most attacks today are classified as class 3 attacks, which means that either the costs associated to break the system are far more than the cost of the system itself, or that the cracker has to spend several or hundred years of computing power to break into a single transaction. Technology is developing faster than cracker methods. Security Level concept is very important to reduce the burden of security countermeasure installation. But in this paper, the aspect related to economy is not included because of too many variables, and this work should be done in near future. Therefore, each new generation of technology usually prevents attacks that the previous generation was vulnerable to.

References

[1] Tamas Vilaghy, Vasukh Doddaballapur, Sebastien Llaurency Hong Min Dinkar Tiwari October 2002. Writing Optimized Java Applications for z/OS

[2] Patrick Kappeler, Jonathan Barney, Pierre Béda, Michael Buzzetti, Saheem Granados, Ebbe Mølgaard Pedersen, Kin Ng, Michael Onghena, Eysha Powers, Martina Schmidt, Richard Schultz December 2008. Java Security on z/OS - The Complete View

[3] Security Target for IBM z/OS Version 1Release 10 *Version 5.11 March 16, 2009*

[4] Tai-Hoon Kim, Gil-cheol Park and Seok-soo Kim ”OS Security Enhancement System by Considering Security Level” International Journal of Multimedia and Ubiquitous Engineering Vol. 2, No. 4, October, 2007.

[5] Security Planning and setting up system security *Version 6 Release 1*.